1. FSM

1-1. Player

1-2. Monster

2. 땅따먹기 로직

3. 링크드 리스트

그림

플레이어 state 변화

휴식

땅따먹기 준비

땅따먹기 하는중

돌아가기

몬스터 state 변화

땅따먹기 로직

1. 기획 내용

땅따먹기와 포켓몬과의 배틀을 컨셉으로 포켓몬의 공격을 회피 및 방어 하면서 보유한 포켓몬의 스킬을 활용하여 80% 이상의 땅을 모두 점령하면 승리하는 게임

1-1. 플레이어

1-1.1 플레이어 상태

플레이어는 위치와 점령 상태에 따라 4가지 상태를 가지게 된다.

* ON\_OCCPIED  
  플레이어가 점령된 땅에서 돌아다닐 수 있으며 몬스터와 충돌되지 않는다.
* READY\_OCG  
  플레이어가 점령된 땅에서 점령을 시작하기 전으로 점령된 땅 밖으로 나갈 수 있는 상태다.
* OCCUPYING  
  플레이어가 점령된 땅을 나와 새롭게 점령하고 있는 상태
* GOBACK  
  플레이어가 새롭게 점령하고 있는 상태에서 더 이상 점령을 하지 않고 점령한 역 방향으로 자동으로 돌아가는 상태

텍스트, 스크린샷, 폰트, 직사각형이(가) 표시된 사진

자동 생성된 설명

1-1.2 플레이어 달리기 및 체력

Shift를 눌러 달리기를 할 수 있으며 체력이 소모된다. 체력은 플레이어 머리 위에 위치하며 플레이어 상태에 따라 소모 속도와 충전 속도가 상이하다. 

플레이어 이동은 점령 중 상태가 아닐 시 점령된 선들을 따라 이동할 수 있다. 새롭게 점령할 때는 점령된 땅을 벗어나 파란색 라인으로 점령중인 선이 표시된다.

A screenshot of a video game

Description automatically generated 스크린샷, 만화 영화, 애니메이션, 그래픽 디자인이(가) 표시된 사진

자동 생성된 설명

\* 점령중인 땅 이동 \* 새롭게 점령중인 땅 이동

1-2.소유 몬스터 (피카츄)

1-2.1썬더볼트

Q를 Hold하여 플레이어 중심으로 포켓몬을 조종하며, Hold시간과 비례하여 투사체 개수가 최대 10개 까지 충전된다. Q를 Away 시 피카츄 중심으로 랜덤한 방향으로 투사체가 발사된다. 스킬 대기시간은 피카츄 머리 위에 표시된다.

몬스터의 투사체와 충돌 시 두 투사체는 사라진다. 몬스터와 충돌 시 몬스터 이동 속도가 감소되고, 제한된 시간 안에 5번을 맞추면 몬스터가 마비로 전환된다.

스크린샷, 만화 영화, 그래픽 디자인, 애니메이션이(가) 표시된 사진

자동 생성된 설명 스크린샷, 만화 영화, 그래픽 디자인, 그린이(가) 표시된 사진

자동 생성된 설명

\* 썬더볼트 장전 \* 썬더볼트 발사

1-2.2 100만 볼트

썬더볼트와 동일한 기능의 투사체가 100개 생성된다. R을 눌러 즉시 사용 가능하며 재사용 시간이 존재하지 않지만 사용 개수가 존재한다.

A screenshot of a video game

Description automatically generated

\* 100만 볼트 발사

1-2. 몬스터

몬스터는 점령되지 않은 땅에서 벽들과 충돌하면서 돌아다니며 상태는 총 3가지 상태가 존재한다



1-2.2 몬스터 공격

- 투사체 공격

몬스터가 돌아다니며 투사체를 놓고가며 천천히 한 방향으로 가다 바로 빠르게 발사된다. 플레이어가 점령상태 때 점령 중인 점들 사이 파란색 선과 충돌시 플레이어의 목숨이 깎이며 플레어는 점령 시작 점으로 복귀된다.

- 늪지대 공격

몬스터는 플레이어 위치에 늪지대를 생성하여 플레이어 이동속도를 저하 시킨다.

일러스트레이션, 만화 영화, 클립아트, 애니메이션이(가) 표시된 사진

자동 생성된 설명

\* 몬스터 투사체, 늪지대 공격

핵심 코드

1. FSM

다이어그램

텍스트, 스크린샷, 원, 폰트이(가) 표시된 사진

자동 생성된 설명

1. 플레이어가 점령된 땅에서 점령 키를 PRESS하여 준비
2. 점령 준비 상태에서 점령된 땅을 나와 새롭게 점령하는 상태(점령 키 HOLD)
3. 새롭게 점령 중인 상태에서 점령 키를 AWAY 하여 새롭게 점령한 점 순서대로 빠르게 복귀
4. 점령 준비 상태에서 점령 키 AWAY
5. 점령한 땅을 다시 만나 새로운 땅 점령 완료
6. 새롭게 점령한 모든 점을 복귀하여 시작점으로 돌아온 상태

플레이어 FSM

void CPlayer::PlayerStateUpdate()

{

switch (eState)

{

case PLAYER\_STATE::ON\_OCCUPIED:

StateOnOccupied();

break;

case PLAYER\_STATE::READY\_OCG:

StateReadyOccupying();

break;

case PLAYER\_STATE::OCCUPYING:

StateOccupying();

break;

case PLAYER\_STATE::GOBACK:

StateGoBack();

break;

}

}

ON\_OCCUPIED State

* 점령된 땅에 있는 상태로 달리기 상관 없이 지속적으로 체력이 찬다.

void CPlayer::StateOnOccupied()

{

if (fStamina < 50)

{

fStamina += fRecoverStamina \* fDT;

if (fStamina > 50)

fStamina = 50;

}

}

READY\_OCG State

* 점령 준비 상태로 점령된 땅을 돌아다니며 동일 State를 유지할 수 있고 점렴되지 않은 땅 방향으로 이동하여 OCCUPYING State 으로 전환될 수 있다.

void CPlayer::StateReadyOccupying()

{

// 플레이어가 점령 준비 상태에서 새롭게 점령을 출발했는지 아니면 아직 점령된 땅을 이동하는지 확인

OCDPoint\* ocdPoint = GetOCDVec2();

OCDPoint\* temp = ocdPoint;

do

{

if (ocdPoint->pPoint.AroundEqual(pCenter))

{

pOCGPoint[0] = ocdPoint->pPoint;

OCGonOCD = true;

break;

}

else

OCGonOCD = false;

ocdPoint = ocdPoint->nxt;

} while (ocdPoint != temp);

if (!OCGonOCD)

pOCGPoint[0] = vCenter;

ocdPoint = ocdPoint->nxt;

bCollect = true;

eDirStart = DIRECTION::END;

}

OCCUPYING State

* 플레이어가 점령된 땅을 벗어나 새롭게 점령하며 방향을 변경할 때 마다 점들을 링크드 리스트 형태로 수집한다 : CollectOCGPoints()
* 시작 위치가 아닌 다른 위치의 점령된 땅을 다시 만나면 점령이 완료되어 링크드 리스트형태의 점령된 땅 사이에 새롭게 점령된 링크드 리스트가 추가 된다.

void CPlayer::StateOccupying()

{

// 새롭게 점령하고 있는 땅들 포인트 수집 하다가 플레이어가 점령된 땅에 다시 닿을 시 수집된 점들 새롭게 추가

if (eDirStart == DIRECTION::END)

{

eDirStart = eDirCrnt;

ocgCnt = 1;

}

CollectOCGPoints();

if (IsPlayerOnOCD())

{

eState = PLAYER\_STATE::ON\_OCCUPIED;

OCDPoint\* ocdCrnt = GetOCDVec2();

OCDPoint\* temp = ocdCrnt;

do

{

if (IsBetweenX(ocdCrnt->pPoint, ocdCrnt->nxt->pPoint, pOCGPoint[ocgCnt]))

{

if (abs(ocdCrnt->pPoint.y - pOCGPoint[ocgCnt].y) <= 1

&& abs(ocdCrnt->pPoint.y - pOCGPoint[ocgCnt].y) >= 0)

pOCGPoint[ocgCnt].y = ocdCrnt->pPoint.y;

}

if (IsBetweenY(ocdCrnt->pPoint, ocdCrnt->nxt->pPoint, pCenter))

{

if (abs(ocdCrnt->pPoint.x - pOCGPoint[ocgCnt].x) <= 1

&& abs(ocdCrnt->pPoint.x - pOCGPoint[ocgCnt].x) >= 0)

pOCGPoint[ocgCnt].x = ocdCrnt->pPoint.x;

}

} while (ocdCrnt != temp);

ocgCnt++;

CScene\* crntScene = CSceneMgr::GetInst()->GetCurrentScene();

crntScene->GetOCD()->FinishOCG(GetOCGPack());

OCGonOCD = false;

ocgCnt = 0;

}

}

GOBACK State

* 플레이어가 해당 State에서 점령된 땅을 만나면 ON\_OCCUPIED State로 전환된다.
* 위 사항이 아니면 마지막으로 새롭게 점령된 점 방향으로 이동하며 만나면 해당 점을 삭제 후 다시 마지막 점으로 이동을 반복한다.

void CPlayer::StateGoBack()

{

bRetreat = true;

if (IsPlayerOnOCD())

{

eState = PLAYER\_STATE::ON\_OCCUPIED;

ocgCnt = 0;

}

else MoveGoBack();

}

void CPlayer::MoveGoBack()

{

Point vLastOcg = pOCGPoint[ocgCnt - 1];

Point v\_iPos = { Vec2(vCenter.x, vCenter.y).ToPoint()};

if (eState == PLAYER\_STATE::OCCUPYING)

{

vCenter = pOCGPoint[ocgCnt].ToVec2();

eState = PLAYER\_STATE::GOBACK;

}

else if (eState==PLAYER\_STATE::GOBACK)

{

if (ocgCnt == 0)

{

vCenter = pOCGPoint[0].ToVec2();

}

if ((int)vLastOcg.x == (int)v\_iPos.x)

{

if ((int)vLastOcg.y == (int)v\_iPos.y)

ocgCnt--;

else if ((int)vLastOcg.y > (int)v\_iPos.y)

MoveDown();

else

MoveUp();

}

else if ((int)vLastOcg.y == (int)v\_iPos.y)

{

if ((int)vLastOcg.x == (int)v\_iPos.x)

ocgCnt--;

else if ((int)vLastOcg.x > (int)v\_iPos.x)

MoveRight();

else

MoveLeft();

}

}

pOCGPoint[ocgCnt] = v\_iPos;

bCollect = false;

}

2. 링크드 리스트

점령된 땅은 링크드 리스트로 각 점들을 순차적으로 연결되어 있다. 새롭게 점령 한 땅은 기존 점령된 땅의 링크드 리스트 중간에 삽입된다. 새로운 링크드 리스트가 삽입 될 때 기존 링크드 리스트 점이 삭제될 수 있다.

기존 링크드 리스트에 신규 링크드 리스트 삽입

스크린샷, 다채로움, 라인, 직사각형이(가) 표시된 사진

자동 생성된 설명

기존 링크스 리스트 점 1과 2 사이에 신규 링크드 리스트 1~4가 삽입되어 최종 점 8개의 링크드 리스트 생성

기존 링크드 리스트 일부 점 삭제 및 신규 링크드 리스트 삽입

스크린샷, 다채로움, 픽셀이(가) 표시된 사진

자동 생성된 설명

기존 링크드 리스트 점 3과 7 사이에 신규 링크드 리스트 1~3가 삽입 되면서 기존 링크드 리스트 4, 5, 6이 삭제 되고 점 8개의 링크드 리스트 생성

점 구조체

* OCDPoint 구조체

헤더파일

struct OCDPoint

{

Point pPoint;

OCDPoint\* nxt;

OCDPoint\* prv;

public:

OCDPoint();

OCDPoint(Vec2 \_vec2);

void InsertNxt(OCDPoint\* \_OCDPoint);

void InsertPrv(OCDPoint\* \_OCDPoint);

void InsertNxt(Vec2 \_vec2);

void InsertPrv(Vec2 \_vec2);

void DeleteSelf();

};

-

정의 파일

OCDPoint::OCDPoint()

:pPoint(0, 0)

, nxt(nullptr)

, prv(nullptr)

{}

OCDPoint::OCDPoint(Vec2 \_vec2)

{

pPoint = \_vec2;

nxt = nullptr;

prv = nullptr;

}

void OCDPoint::InsertNxt(OCDPoint\* \_OCDPoint)

{

\_OCDPoint->prv = this;

\_OCDPoint->nxt = nxt;

nxt->prv = \_OCDPoint;

nxt = \_OCDPoint;

}

void OCDPoint::InsertPrv(OCDPoint\* \_OCDPoint)

{

\_OCDPoint->nxt = this;

\_OCDPoint->prv = prv;

prv->nxt = \_OCDPoint;

prv = \_OCDPoint;

}

void OCDPoint::InsertNxt(Vec2 \_vec2)

{

OCDPoint\* \_OCDPoint = new OCDPoint(\_vec2);

InsertNxt(\_OCDPoint);

}

void OCDPoint::InsertPrv(Vec2 \_vec2)

{

OCDPoint\* \_OCDPoint = new OCDPoint(\_vec2);

InsertPrv(\_OCDPoint);

}

void OCDPoint::DeleteSelf()

{

prv->nxt = nxt;

nxt->prv = prv;

delete this;

}

링크드 리스트 시작 4개 점 설정

void COcuppied::SetStartOCD()

{

RECT rect = CCore::GetInst()->GetOCDTotalSize();

OCDPoint\* temp = new OCDPoint(Vec2(rect.left, rect.top));

fTotalArea = (rect.right - rect.left) \* (rect.bottom - rect.top);

// >> : set head

ocdVec2 = temp;

ocdVec2->nxt = ocdVec2;

ocdVec2->prv = ocdVec2;

// << : set head

// >> : insert start points

ocdVec2->InsertNxt(Vec2(rect.right, rect.top));

ocdVec2 = ocdVec2->nxt;

ocdVec2->InsertNxt(Vec2(rect.right, rect.bottom));

ocdVec2 = ocdVec2->nxt;

ocdVec2->InsertNxt(Vec2(rect.left, rect.bottom));

// << : insert start points

}

신규 링크드 리스트 수집

void CPlayer::CollectOCGPoints( )

{

if (eDirBefore != eDirCrnt && eDirBefore != OppositeDirection(eDirCrnt))

{

pOCGPoint[ocgCnt] = vCenter;

if (abs(pOCGPoint[ocgCnt].x - pOCGPoint[ocgCnt - 1].x) == 1)

{

pOCGPoint[ocgCnt].x = pOCGPoint[ocgCnt - 1].x;

}

else if (abs(pOCGPoint[ocgCnt].y - pOCGPoint[ocgCnt - 1].y) == 1)

{

pOCGPoint[ocgCnt].y = pOCGPoint[ocgCnt - 1].y;

}

ocgCnt++;

}

else

{

pOCGPoint[ocgCnt] = vCenter.ToPoint();

}

}

신규 링크드 리스트 추가 위치 확인 및 삭제/삽입

void COcuppied::FinishOCG(PackOCG \_ocgPack)

{

ocgPack = \_ocgPack;

CScene\* crntScene = CSceneMgr::GetInst()->GetCurrentScene();

vEnemyCen = crntScene->GetEnemy()->GetPosition();

if(!ocgPack.bSameDir)

{

if (!IsEnemyInOCG())

ocgPack.bClockwise = !ocgPack.bClockwise;

ConfigDeletePoint();

}

else

{

OCGSameDirection();

}

if (ocdDeleteStart == nullptr || ocdDeleteEnd == nullptr || ocdAddStart == nullptr) return;

DeleteOCDPoint();

AddOCGPoint();

bStartOCGonOCD = false;

bEndOCGonOCD = false;

StraightOCD();

UpdateOCDArea();

if (fTotalPercent> FinishRatio)

{

bFinishGame = true;

fDTFinishGame = 0;

}

}

몬스터가 신규 링크드 리스트 안에 있는지 밖에 있는지 확인

bool COcuppied::IsEnemyInOCG()

{

int enemyLineCnt = 0;

//몬스터 좌우위아래 중 ocg 선이 홀수면 in 짝수면 out

DIRECTION startCnt = DIRECTION::END;

for (int i = 0; i < ocgPack.ocgCnt; i++)

{

if ((startCnt == DIRECTION::END || startCnt == DIRECTION::RIGHT) && vEnemyCen.x < ocgPack.ocgPoint[i].x)

{

if (IsBetweenY(ocgPack.ocgPoint[i], ocgPack.ocgPoint[(i + 1)%ocgPack.ocgCnt], vEnemyCen))

{

enemyLineCnt++;

startCnt = DIRECTION::RIGHT;

}

}

if ((startCnt == DIRECTION::END || startCnt == DIRECTION::LEFT) && vEnemyCen.x > ocgPack.ocgPoint[i].x)

{

if (IsBetweenY(ocgPack.ocgPoint[i], ocgPack.ocgPoint[(i + 1) % ocgPack.ocgCnt], vEnemyCen))

{

enemyLineCnt++;

startCnt = DIRECTION::LEFT;

}

}

if ((startCnt == DIRECTION::END || startCnt == DIRECTION::DOWN) && vEnemyCen.y < ocgPack.ocgPoint[i].y)

{

if (IsBetweenX(ocgPack.ocgPoint[i], ocgPack.ocgPoint[(i + 1) % ocgPack.ocgCnt], vEnemyCen))

{

enemyLineCnt++;

startCnt = DIRECTION::DOWN;

}

}

if ((startCnt == DIRECTION::END|| startCnt == DIRECTION::UP) && vEnemyCen.y > ocgPack.ocgPoint[i].y)

{

if (IsBetweenX(ocgPack.ocgPoint[i], ocgPack.ocgPoint[(i + 1) % ocgPack.ocgCnt], vEnemyCen))

{

enemyLineCnt++;

startCnt = DIRECTION::UP;

}

}

}

if (enemyLineCnt % 2 != 0)

{

return true;

}

return false;

}

기존 링크드 리스트 삭제 점 확인

void COcuppied::ConfigDeletePoint()

{

ocdDeleteStart = nullptr;

ocdDeleteEnd = nullptr;

ocdAddStart = nullptr;

OCDPoint\* temp = ocdVec2;

if (ocgPack.bClockwise)

{

do

{

if (IsOnXorYAxis(ocdVec2->pPoint, ocdVec2->nxt->pPoint, ocgPack.ocgPoint[0])

&&ocdDeleteStart==nullptr)

{

ocdDeleteStart = ocdVec2->nxt;

ocdAddStart = ocdVec2;

}

if (IsOnXorYAxis(ocdVec2->pPoint, ocdVec2->nxt->pPoint, ocgPack.ocgPoint[ocgPack.ocgCnt - 1]))

{

ocdDeleteEnd = ocdVec2;

}

if (ocdVec2->pPoint.AroundEqual(ocgPack.ocgPoint[0]))

{

ocdDeleteStart = ocdVec2;

ocdAddStart = ocdVec2->prv;

}

ocdVec2 = ocdVec2->nxt;

} while (temp != ocdVec2);

}

else //ocgPack.bClockwise == true

{

do

{

if (IsOnXorYAxis(ocdVec2->pPoint, ocdVec2->nxt->pPoint, ocgPack.ocgPoint[0])

&&ocdDeleteStart==nullptr)

{

ocdDeleteStart = ocdVec2;

ocdAddStart = ocdVec2->nxt;

}

if (IsOnXorYAxis(ocdVec2->pPoint, ocdVec2->nxt->pPoint, ocgPack.ocgPoint[ocgPack.ocgCnt - 1]))

{

ocdDeleteEnd = ocdVec2->nxt;

}

if (ocdVec2->pPoint.AroundEqual(ocgPack.ocgPoint[0]))

{

ocdDeleteStart = ocdVec2->prv;

ocdAddStart = ocdVec2;

}

ocdVec2 = ocdVec2->nxt;

} while (temp != ocdVec2);

}

}

기존 링크드 리스트 점 삭제

void COcuppied::DeleteOCDPoint()

{

OCDPoint\* temp = ocdDeleteStart;

//시작점과 끝점 사이에 아무 점이 없고 몬스터가 안에 없을 때

if (ocdDeleteStart == ocdDeleteEnd&&!ocdDeleteStart->pPoint.AroundEqual(ocdDeleteEnd->pPoint))

{

ocdDeleteStart->DeleteSelf();

ocdDeleteStart = nullptr;

}

//시작점과 끝점 사이에 아무 점이 없고 몬스터가 안에 있을 때

else if (IsEnemyInOCG()&&(ocdDeleteStart == ocdAddStart || ocdDeleteEnd == ocdAddStart))

{

if(ocgPack.bClockwise)

{

while (temp != ocdDeleteEnd)

{

temp = ocdDeleteStart->nxt;

ocdDeleteStart->DeleteSelf();

ocdDeleteStart = temp;

}

ocdDeleteEnd->DeleteSelf();

}

else //ocgPack.bClockwise==false;

{

while (temp != ocdDeleteEnd)

{

temp = ocdDeleteStart->prv;

ocdDeleteStart->DeleteSelf();

ocdDeleteStart = temp;

}

ocdDeleteEnd->DeleteSelf();

}

bDeleteAllPoints = true;

}

else if (ocgPack.bClockwise)

{

OCDPoint\* ocdEndNxt = ocdDeleteEnd->nxt;

while(temp!=ocdEndNxt)

{

temp = temp -> nxt;

ocdDeleteStart->DeleteSelf();

ocdDeleteStart = temp;

}

}

else //ocgPack.bClockwise==false

{

OCDPoint\* ocdEndPrv = ocdDeleteEnd->prv;

while (temp != ocdEndPrv)

{

temp = temp->prv;

ocdDeleteStart->DeleteSelf();

ocdDeleteStart = temp;

}

}

}

신규 링크드 리스트 점 추가

void COcuppied::AddOCGPoint()

{

int i = 0;

if (bStartOCGonOCD) i = 1;

if (bEndOCGonOCD) ocgPack.ocgCnt--;

if (!bDeleteAllPoints)

{

if (ocgPack.bClockwise)

{

for (; i < ocgPack.ocgCnt; i++)

{

ocdAddStart->InsertNxt(ocgPack.ocgPoint[i].ToVec2());

ocdAddStart = ocdAddStart->nxt;

}

}

else //ocg.PackbClockwise==false

{

for (; i < ocgPack.ocgCnt; i++)

{

ocdAddStart->InsertPrv(ocgPack.ocgPoint[i].ToVec2());

ocdAddStart = ocdAddStart->prv;

}

}

ocdVec2 = ocdAddStart;

}

else

{

ocdVec2= new OCDPoint(ocgPack.ocgPoint[2]);

ocdVec2->prv = new OCDPoint(ocgPack.ocgPoint[1]);

ocdVec2->nxt = ocdVec2->prv;

ocdVec2->prv->prv = ocdVec2;

ocdVec2->prv->nxt = ocdVec2;

ocdVec2->InsertNxt(ocgPack.ocgPoint[0]);

for (int i = 3; i < ocgPack.ocgCnt; i++)

{

ocdVec2->InsertNxt(ocgPack.ocgPoint[i].ToVec2());

ocdVec2 = ocdVec2->nxt;

}

bDeleteAllPoints = false;

}

}

점령된 땅 업데이트

void COcuppied::UpdateOCDArea()

{

OCDPoint\* temp = ocdVec2;

float fSum1 = 0;

float fSum2 = 0;

do

{

fSum1 += ocdVec2->pPoint.x \* ocdVec2->nxt->pPoint.y;

fSum2 += ocdVec2->pPoint.y \* ocdVec2->nxt->pPoint.x;

ocdVec2 = ocdVec2->nxt;

} while (ocdVec2 != temp);

fOCDArea = (fSum1 - fSum2) / 2;

fTotalPercent = (fTotalArea - fOCDArea) / fTotalArea;

}

3. 땅따먹기 로직